(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 November 2001 (29.11.2001)

PCT

(10) International Publication Number
WO 01/91039 A1

(51) International Patent Classification⁷: G06K 9/36, H04N 7/12

(21) International Application Number: PCT/US01/16579

(22) International Filing Date: 23 May 2001 (23.05.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/576,700 23 May 2000 (23.05.2000) US

(71) Applicant: COMPRESSION ENGINE TECHNOLOGIES, INC. [US/US]; 30 Persimmon Place, Hilton Head Island, SC 29926-2659 (US).

(72) Inventor: WANG, Jianrong; 46 S. Dreamweaver Circle, The Woodlands, TX 77380 (US).

(74) Agent: ANDERSON, Rodney, M.; Anderson, Levine & Lintel, L.L.P., Suite 111, 12160 Abrams Road, Dallas, TX 75243 (US).

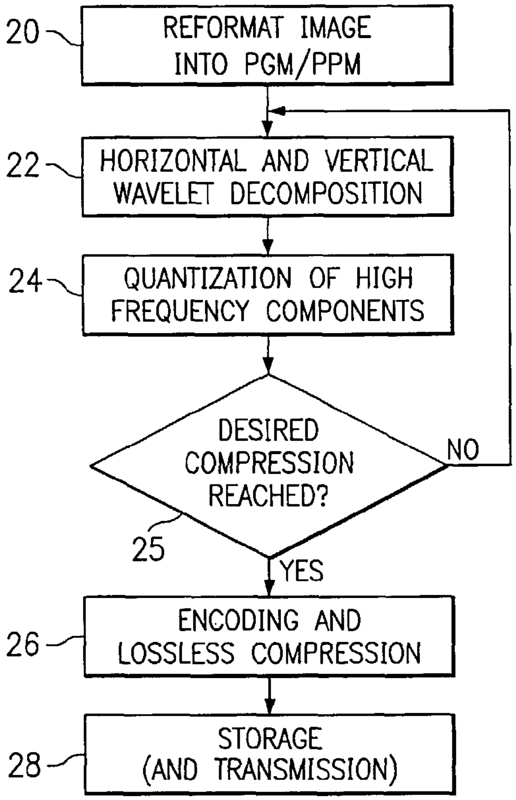
(81) Designated States *(national)*: CA, JP.

(84) Designated States *(regional)*: European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR).

Published:
— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: MEMORY-EFFICIENT IMAGE COMPRESSION


20
REFORMAT IMAGE INTO PGM/PPM
22
HORIZONTAL AND VERTICAL WAVELET DECOMPOSITION
24
QUANTIZATION OF HIGH FREQUENCY COMPONENTS
DESIRED COMPRESSION REACHED?
NO
25
YES
26
ENCODING AND LOSSLESS COMPRESSION
28
STORAGE (AND TRANSMISSION)

(57) Abstract: The disclosed method and system (8) for image compression includes line buffer (52) in cache memory (14, 18) horizontal and vertical wavelet decomposition (22), cache memory (14, 18) having line buffers (52) wherein the number of line buffers (52) corresponds to the length of the vertical wavelet filter, pointers (53), quantization (24), encoder (26), and storage and transmission (28).

MEMORY-EFFICIENT IMAGE COMPRESSION

* * * * *

BACKGROUND OF THE INVENTION

This invention is in the field of digital communication of visible images, and is more specifically directed to wavelet transform image compression and encoding techniques.

In recent years, the ever-increasing availability of high-performance computer systems has greatly changed the manner in which visual images are generated, stored, and communicated. Visual images are now commonly digitally captured by digital cameras, digitally generated on a computer in original form or as modifications of previous images, or digitally scanned from paper copies. The advent of inexpensive disk drives, with many gigabytes of capacity, facilitates storage of many large full-color images. In the area of communications, the explosive growth in the use of the Internet and significant increases in communication bandwidth, have greatly increased the popularity of communication of digital images among computer users. Indeed, a large majority of currently accessible Internet websites are not text-only, but rely upon the display of images. Individuals now commonly communicate favorite snapshots and other images to friends and family by email. Internet commerce also heavily utilizes the transmission and display of digitally stored still images in online catalogs, product descriptions, advertisements, and website ornamentation.

These recent increases in communications bandwidth and computer storage capacity have also increased the demand for the storage and communication of visual images. Accordingly, many approaches have been developed to compress the digital data representations of visual images, so that fewer bits are required to store and communicate the corresponding image, preferably with minimal distortion in the image.

Conventional data compression techniques are referred to as either of the lossless or lossy type. The defining characteristic of lossless compression is the ability to recover an exact copy of the input information, upon decompression. Because of this feature, lossless compression is useful in the storage and communication of computer programs, numerical databases, and other information in which exact replication is a requirement. Lossy compression, on the other hand, involve some approximation of the original image, and therefore the reconstructed information cannot exactly match that of the original information; in other words, some information is lost in the compression of the input information. Lossy compression techniques can provide very high compression ratios, however, and as such are often used when the information to be compressed does not require exact bit-for-bit replication upon decompression. As a result, lossy data compression techniques are useful in the compression of video or graphics images, audio signals, and other digital representation of analog information.

In general, lossless data compression typically combines repetitive bits in the digital representation of information. An example of lossless compression is commonly referred to as "byte packing". Run Length Encoding (RLE) is another conventional lossless compression method that is particularly well-suited for raw image data in the case where data retention is critical. The RLE algorithm compresses images by condensing strings of identical pixel values into a first byte that identifies the number of pixels in the string, and a second byte that records the pixel value for the string. Raw image data is particularly well-suited for RLE compression, as large areas of a constant color or brightness are efficiently encoded according to this approach.

In any case, the compression ratio achieved from modern lossless compression is relatively low. Indeed, real-time communication of the sequence and significant storage savings are not possible if only lossless compression is applied to massive data representations, such as large still images, and sequences of images such as motion pictures and seismic surveys. However, lossless compression is often used in combination with lossy data compression schemes, for example in further compressing an image that is compressed by a lossy technique.

Typically, lossy data compression techniques may be considered as low-pass filters of the input information. High frequency effects in graphics images generally correspond to sharp edges of shapes in the image. The application of lossy compression to these images is generally reflected as a loss of resolution at the edges of image elements, or by a loss of resolution in those portions of the image that change rapidly over time or space;

the low-frequency portions, however, can provide a substantially accurate image upon decompression.

In order to obtain accurate decomposition in compression, many relatively complex transformation processes are known in the art. One type of decomposition, commonly referred to as JPEG (Joint Photographic Experts Group) compression, divides each image into blocks, and applies a Discrete Cosine Transform to each block to produce arrays of coefficients which are then quantized and subjected to a difference algorithm. Another approach, referred to as fractal compression, divides each image into pixel groups, or tiles, and approximates each tile by either or both a contractive or rotational transformation based upon a reference region of the image, producing a compressed image consisting of a full representation of the reference region plus the transformation operators for each tile.

Wavelet-based compression techniques are also known in the art, and have been applied to both still images and motion pictures. U.S. Patent No. 5,604,824, incorporated by reference hereinto, describes several wavelet-based approaches to the compression of still images, including the use of dual basis wavelet functions, interpolatory wavelets, and wavelet packets. U.S. Patent No. 5,600,373, also incorporated by reference hereinto, describes the use of boundary-spline-wavelet function pairs to compress and decompress motion pictures and other sequences of images. Other wavelet transforms that are commonly used for image compression are described in Antonini, et al., "Image coding using the wavelet transform", *IEEE Trans. Image Proc. 1* (April, 1992), pp. 205-220, also incorporated herein by this reference.

Typically, these filter-based lossy compression techniques, whether of the DCT, fractal, or wavelet type, filter the input image into low and high frequency components. This filtering is then followed by a quantization operation, in which the filtered data is set to zero for values below a quantization threshold. Since most real-world images are dominated by the low-frequency information, the high frequency components can be very highly compressed according to this technique, particularly as the decomposition is repeated over a number of levels.

The decomposition of images is conventionally carried out sequentially in the horizontal and vertical image directions. For example, as described in the abovereferenced U. S. Patents No. 5,604,824 and No. 5,600,373, the compressed image is decompressed first in the horizontal direction to generate a low frequency component and a high frequency component. This decomposition is illustrated in Figure 1, which illustrates memory buffer 2 that contains the digital data corresponding to the image to be compressed. In the conventional manner, these digital data are expressed as a digital value for each picture element, or picture, of the image. The size of the digital data element for each pixel can range from one bit for a black-and-white image, to one or two bytes per pixel for a gray scale image, to multiple bytes per pixel for a full-color image. The digital pixel data are initially arranged in memory buffer 2 according to the physical position the input image.

The input image is first decomposed, row-by-row, in the horizontal direction. The result of this horizontal decomposition is a low-frequency component and a high-frequency component, each component having half the number of samples as the original image. These components are then stored in memory in a horizontal orientation, for example replacing the original image row. Upon completion of this horizontal decomposition for each row of the image, memory buffer 2′ stores the image arranged in its the low-frequency and high-frequency components, split within rows as shown in Figure 1.

These two components are then vertically decomposed using the same filter approach, column-by-column, with the resulting low and high frequency components for each column stored in the same memory region, in the same vertical column. As a result of this bidirectional decomposition, memory buffer 2″ stores four components, designated as low-low (LL), high-high (HH), high-low (HL), and low-high (LH) to indicate the horizontal and vertical decomposition components, respectively, in the resultant component.

It has been observed, however, that the execution of this bidirectional approach by a conventional microprocessor-based computer requires a significant number of accesses to system memory. According to conventional techniques, the entire image memory is scanned through at least two times, once for the horizontal decomposition and then again for the vertical decomposition of the horizontal results.

Referring to Figure 2, consider the example of memory buffer 2 of dimensions H by W. For memory buffer 2 that is stored in a sequential buffer, pixel data x(m,n), corresponding to the pixel in the mth row and the nth column of the image, is stored at a memory address offset of nW+m from the base address of pixel x(0,0). Horizontal decomposition is then applied to the image data. In this example, vertical decomposition of column m requires retrieval of data from memory buffer 2 at offset m in each row, beginning with the first, or zeroth, row. In other words, data must be retrieved from the n buffer locations specified by k * W + m, for k ranging from 0 to H-1 (i.e., for each of the H rows). According to these conventional techniques, therefore, the entirety of memory buffer 2 (and 2′, 2″) is effectively accessed almost W times in the horizontal and vertical decomposition processes.

Considering the relatively large number of memory accesses required in this bidirectional decomposition, a large number of central processing unit (CPU) operations are directed to these operations. Of course, accesses to memory external to the CPU can each occupy several CPU cycles, due to the access time of external memory. As a result, external memory accesses involved in the decomposition process can occupy a significant amount of the overall compression processing time, and indeed may be a limiting factor in the overall compression rate and compression ratio.

As is known in the computer architecture art, cache memories are now often used to provide rapid access by the CPU to frequently-accessed memory locations. Because of their high performance, these cache memories are relatively expensive, on a cost-per-bit basis, and as such are generally quite small in conventional computer systems. On the other hand, typical still images are represented by a relatively large data block, well past the capacity of typical cache memories. Because of the image range required for vertical decomposition described above, and because of the excessive expense required to construct a cache of sufficient size to rapidly handle the decomposition processes, conventional techniques are unable to efficiently load and utilize reasonably-sized cache memories for this process. Conventional image compression thus typically relies heavily on external memory for image and decomposition component storage.

The eventual results of image decomposition through multiple levels, are quantized, as discussed above, and the quantized results are encoded for storage or communication. In order to save storage capacity and especially communications bandwidth, it is of course desirable to encode these results in as few a number of bits as possible.

By way of further background, a method of efficiently encoding digital bitstreams is described in Rice, et al., "Algorithms for a very high speed universal noiseless coding module", *JPL Publication 91-1,* (Jet Propulsion Laboratory, 1991). According to this approach, a bitstream representative of a sequence of integer values is encoded according to a split sequence approach. This split sequence technique was observed, by the authors of this paper, to provide high performance coding.

BRIEF SUMMARY OF THE INVENTION

It is therefore an object of the present invention to provide a method and apparatus for performing image compression while requiring a relatively small amount of active memory for storage of input data and results.

It is a further object of the present invention to provide such a method and apparatus that is particularly well-suited for wavelet transforms.

It is a further object of the present invention to provide such a method and apparatus in which the decomposition executes very rapidly relative to conventional methods.

It is a further object of the present invention to provide such a method and apparatus in which the decomposition may be performed with a reduced number of memory accesses.

Other objects and advantages of the present invention will be apparent to those of ordinary skill in the art having reference to the following specification together with its drawings.

The present invention may be implemented in the operation of a central processing unit (CPU) based computer system utilizing external memory to store the overall input image data and decomposition results, in combination with cache memory buffers for storing intermediate results. Buffers in the cache memory are organized to correspond to a number of rows or lines of the input image, with the number of rows equaling the length of the particular filter, for example a wavelet filter. In operation, each line buffer is loaded with a row of the input image, and horizontal decomposition is carried out on a row-by-row basis with these buffers. An ordered sequence of pointers is provided, each pointer pointing to one of the line buffers to indicate its order in the vertical decomposition. Following the horizontal decomposition of these rows, vertical decomposition is carried out for each column position within the line buffers, and the results stored in an output results buffer associated with each decomposition component. The line buffers associated with the earliest two rows in the image are then overwritten with the next two image rows, and the ordered sequence of pointers are rotated. The two new rows are horizontally decomposed, and the vertical decomposition is repeated, including the newly read and decomposed rows.

According to another aspect of the invention, the output of the decomposition process is quantized, and then encoded to further compress the bitstream for storage or communication. The decomposed image sequence is first converted to positive integer values. Each zero value in sequence is counted, and a two's complement number generated according to the number of zeroes in the sequence. In this manner, all non-zero values are represented by a positively-signed data word or byte, while negatively-signed values represent a sequence of zero values, in number corresponding to the absolute value of the negatively-signed word. These values are then further encoded by dividing their values into amplitude groups, with the most significant bits corresponding to the amplitude group for the value and the remainder of the bits indicating the value within the group.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWING

Figure 1 is a diagram of a memory buffer illustrating bidirectional filter decomposition of an input image according to the prior art.

Figure 2 is a diagram of the memory buffer of Figure 1, illustrating the access of a particular location therein.

Figure 3 is an electrical diagram, in block form, of a computer-based system for image compression according to the preferred embodiment of the invention.

Figure 4 is a flow chart illustrating an overall method of image compression according to the preferred embodiment of the invention.

Figure 5 is a flow chart illustrating a method of wavelet decomposition as executed by the system of Figure 3 according to the preferred embodiment of the invention.

Figures 6a through 6c are memory diagrams of the arrangement of buffers in the cache memory of the system of Figure 3 during various stages of wavelet decomposition according to the preferred embodiment of the invention.

Figure 7 is a flow chart illustrating a method of encoding a bitstream corresponding to compressed image data according to the preferred embodiment of the invention.

Figure 8 is a flow chart illustrating a method of performing split sequence encoding of a bitstream as utilized according to the preferred embodiment of the invention.

DETAILED DESCRIPTION OF THE INVENTION

As will be apparent to those skilled in the art, the present invention may be implemented in computing equipment of various types and architectures. For example, the present invention may be implemented in a conventional personal computer or workstation, according to well-known architectures. Alternatively, it is contemplated that the present invention may be implemented in a wide array of digital systems, including such systems as digital image capture systems, handheld and other portable computing devices, dedicated systems for specific applications such as police fingerprint capture and communication, and the like; in particular, it is contemplated that the continuing trend toward miniaturization of electronic components will continue to enable the implementation of image compression techniques into smaller and more portable systems, in addition to conventional desktop workstations. Accordingly, it is to be understood that the following description is presented by way of example only, and is not to be interpreted to limit the scope of the present invention as claimed.

Referring now to Figure 3, the general construction of a computer system capable of performing image compression according to the preferred embodiment of the invention will now be described. In this example, system 8 is a microprocessor-based computer system, such as may be deployed as a personal computer workstation. System 8 includes central processing unit (CPU) 10, which in this example is a general purpose microprocessor of reasonably high performance, having a capability such as the PENTIUM class of microprocessors available from Intel Corporation. Alternatively, it is contemplated that CPU 10 may be realized as an application-specific processor, such as a digital signal processor or graphics processor device, depending upon the particular application of system 8.

As shown in Figure 3, CPU 10 is in communication with several memory resources, of varying access performance and size. Internal cache 12 is a memory resource that is resident within the same integrated circuit as CPU 10 itself. As such, internal cache 12 provides a memory resource that may be accessed very rapidly in the execution of an instruction by CPU 10; for example, it is contemplated that internal cache 12 may, in some cases, be both addressed and written or read within a single machine cycle of CPU 10.

Because of its on-chip implementation, however, internal cache 12 is generally relatively small (e.g. 128 kbyte or smaller) as compared to other memory resources.

A next level of cache memory is provided by backside cache memory 14. Backside cache 14 is referred to as "backside" in that it is in communication with CPU 10 by way of a dedicated cache bus CBUS. This dedication of cache bus CBUS to the purpose of memory access between CPU 10 and backside cache 14 permits its data width to be greater, and its clock rate to be higher, than other general purpose buses connected to CPU 10. As a result, it is contemplated that many cache accesses to backside cache 14 may be accomplished in one or two CPU cycles. Additionally, backside cache 14 is preferably implemented as high speed memory, such as static random access memory (SRAM); because of the cost per bit of such high speed memory, backside cache 14 is also generally relatively small (e.g., 1 Mbyte or smaller).

The next memory level in system 8 is main external memory 18. Main external memory 18 is a relatively large semiconductor memory in this embodiment of the invention, typically having a capacity on the order of megabytes in size, for example 128 Mbytes or larger. Because of this large size, main external memory 18 is preferably implemented by way of dynamic random access memory (DRAM), which has a relatively low cost per bit relative to high-speed SRAM memory. According to this embodiment of the invention, main external memory 18 is coupled to CPU 10 by way of a general purpose bus BUS. Other system functions are also resident on bus BUS, including bulk data storage such as disk memory 19 (which serves as yet another level of memory in system 8). Because of both the relatively slow access times provided by main external memory 18 implemented as DRAM memory, and also latency caused by bus traffic and a relatively slow bus clock rate on bus BUS, data access to main external memory 18 is generally much slower than that to the cache memories 12, 14 in system 8, generally requiring multiple CPU cycles to accomplish.

System 8 also includes numerous system functions, which are not shown in Figure 3 for clarity, depending upon the particular application. For example, system 8 will typically include input functions such as input devices (scanner, digital camera, keyboard, pointing device, etc.), output devices (video display, printers), and data communications functions (modem, network adapters, etc). Generally, these system functions will also reside on general purpose bus BUS along with main external memory 18 and disk memory 19; alternatively, some of the functions may reside upon a separate bus such as is typical in systems utilizing such bus technologies as PCI and USB.

According to the present invention, one or more of these system input functions of system 8 will serve as a source of digital image data, upon which the compression method of the present invention will operate. Accordingly, the digital image to be compressed may be a newly captured image input from a digital camera or scanner, a previously captured image stored in a non-compressed form (e.g., bit map) on disk memory 19, or an image file communicated to system 8 over a network. According to the preferred embodiment of the invention, the image to be compressed may come from these and any other such image source.

In its general operation, digital images are compressed by system 8 according to the preferred embodiment of the invention in a manner following a process flow, such as illustrated in Figure 4 for the example of a gray scale monochrome still image. Of course, those skilled in the art having reference to this specification will readily be able to apply the present invention to other images, including color still images and sequences of images. The following description is directed to the example of a gray scale image for simplicity of the description.

The compression method according to this preferred embodiment of the invention begins with process 20, in which the data representing the digital image are reformatted into a format that is suitable for compression, if the data are not already in such a format. In this example of the compression of a gray scale image, an appropriate format is Portable Grey Map (PGM), in which each pixel of the image is associated with a digital value corresponding to its brightness. For example, eight-bit PGM values range from 0 (black) to 255 (white). For the case of color images, a preferred format output by process 20 is Portable Pixel Map (PPM), in which each of the R, G, and B components of the color image is expressed in a PGM format. Each RGB-PGM pixel is thus associated with three brightness values, one for each of the component colors red, green, and blue.

Once the digital image data are formatted into the desired PGM/PPM format in process 20, horizontal and vertical wavelet decomposition process 22 is then performed upon the image data. Wavelet decomposition process 22 according to the preferred embodiment of the invention, similar to conventional wavelet decomposition techniques, operates in multiple passes, with each pass generating a low-frequency and a high-frequency component. For two-dimensional image data, the decomposition is applied first in one direction, and then in the orthogonal direction. For example, a horizontal decomposition is first performed, generating low and high frequency components upon which vertical decomposition is then performed, as described above relative to Figure 1. Of course, the image may be equivalently decomposed first in the vertical direction and then horizontally.

In general, the output of the initial pass of the horizontal and vertical wavelet decomposition process 22 generates four components, corresponding to the low and high frequency components from the vertical decomposition, as applied to each of the low and high frequency components from horizontal decomposition. The result of this pass of process 22 thus provides the four components LL, HL, LH, HH, with the L and H indicators corresponding to their frequency component output from the horizontal and vertical decomposition passes, respectively. For example, the HL component corresponds to the low-frequency vertical decomposition output of the high-frequency horizontal decomposition component.

Referring back to Figure 4, each component that contains high-frequency information (i.e., the LH, HL, and HH components) are quantized in process 24. Quantization process 24 effectively compares each pixel value of these components to a threshold value, and sets the pixel value to zero if the pixel value is below the threshold; values that are above the threshold are preferably rounded in process 24. The quantization of only the high frequency components from process 22, while lossy, generally only minimally affects the fidelity of most real-world images, as typical visual images are dominated by low-frequency information. Quantization process 24 thus results in a relatively sparse data set, as most high-frequency pixel positions are quantized to zero. However, important abrupt changes or edges in the image will generate a significant high-frequency value that will continue to be represented (although approximated) following quantization process 24 . Quantization process 24 thus greatly reduces the memory requirements for the storage of the image data, while preserving important features.

The low-low (LL) frequency component, which is the low-frequency output component from the vertical decomposition of the low-frequency component of the horizontal decomposition in process 22, is not quantized according to the preferred embodiment of the invention. Rather, decision 25 is performed by system 8 to determine whether the desired compression ratio has been reached. The determination of decision 25 may simply count passes to determine whether a preselected compression ratio has been reached. Alternatively, decision 25 may be dynamically determined through analysis of the quantization results, terminating compression passes upon detecting significant high-frequency information, corresponding to the difference between the LL components in successive decomposition passes. For example, a numerical figure of merit may be derived from the high-frequency information and compared against an error limit. This dynamic compression allows simple images containing primarily low-frequency information to be compressed to a higher ratio than more complex, finely detailed images. In either event, if the desired compression ratio has not yet been reached (decision 25 is NO), control returns back to process 22, where the most recent LL component is again horizontally and vertically decomposed to achieve further compression.

Wavelet-based filters are known in the art to be particularly useful in image compression, because they are capable of localizing information in both time and frequency. Numerous wavelet functions and corresponding filters are known in the art. Examples of wavelet filters that may be applied in process 22 according to the preferred embodiment of the invention are described in U.S. Patent No. 5,604,824, in U.S. Patent No. 5,600,373, and in Antonini, et al., "Image coding using the wavelet transform", *IEEE Trans. Image Proc. 1* (April, 1992), pp. 205-220, all incorporated herein by this reference. It is contemplated that any wavelet-based filter, and indeed any filter that may be applied with finite support, may be used in connection with the present invention.

Referring now to Figure 5, in combination with Figures 6a through 6c, the operation of process 22 according to the preferred embodiment of the present invention will now be described in detail. This description will begin with discussion of the arrangement of high-speed local cache memory, as shown in its initial condition in Figure 6a.

Figure 6a illustrates a memory arrangement that includes a number of line buffers 52_1 through 52_K, for storing input image data and intermediate decomposition results. In this regard, line buffers 52 will be accessed relatively frequently during the execution of process 22. With reference to Figure 3, system 8 preferably implements line buffers 52 in one of the cache memory resources, such as internal cache 12 or backside cache 14, rather than in main external memory 18 (or, of course, in disk memory 19). According to this preferred embodiment of the invention, each line buffer 52 has a width W corresponding to the width of a row of the input image, so that each line buffer 52 can store a complete row of the image. Considering that the size of input images will vary from one another and as additional compression passes are performed, it is contemplated that line buffers 52 will be dynamically arranged in the appropriate one of cache memories 12, 14.

As noted above, the number of line buffers 52 that are allocated for an image compression operation is set to a value K. According to this preferred embodiment of the invention, K is the length of the wavelet filter that will be applied to the image data in the vertical decomposition operation in the image compression process. For example, the Daubechies 9/7 wavelet-based filter described in Antonini, et al., "Image coding using the wavelet transform", *IEEE Trans. Image Proc. 1* (April, 1992), pp. 205-220, incorporated by reference above, provides a filter length K of 9, in which case nine line buffers 52_1 through 52_9 will be allocated. While typically the length of the wavelet filter is the same in each of the horizontal and vertical decomposition processes, it is contemplated that different wavelets or filters may be used in the horizontal and vertical directions; according to this embodiment of the invention, the number K of line buffers corresponds to the length of the filter applied in the vertical direction.

Each line buffer 52 is associated with an entry in a set of pointers 53, which are also stored in one of the cache resources 12, 14 or in a register file within CPU 10 in system 8. Each pointer 53 stores a value IPtr() that indicates the order in which the contents of its associated line buffer 52 is to be processed in vertical decomposition, as will be described below. As such, the sets of values IPtr() vary from 1 to K, in sequence, with these values associated with line buffers 52 in a wrap-around manner as will be apparent from the following description.

Referring now to Figure 5, the image compression process executed by CPU 10 in system 8 according to the preferred embodiment of the invention begins with process 30, in which line buffers 52 are loaded with data corresponding to rows of the input image. As noted above, K line buffers 52 are provided within cache memory 12 or 14, and as such process 30 scans K rows of the input image into line buffers 52_1 through 52_K. As such, a portion of the input image is now ready for wavelet transformation.

In process 32, CPU 10 assigns the initial values IPtr() stored by pointers 53 to match the initial allocation of line buffers 52, so that the pointer value IPtr(1) points to line buffer 52_1, which stores the first row of the input image; pointer value IPtr(2) points to line buffer 52_2, which stores the second row of the input image; and so on, with pointer value IPtr(K) pointing to line buffer 52_K, which stores the K^{th} row of the input image. The state of line buffers 52 and pointers 53 after completion of initial assignment process 32 is illustrated in Figure 6a.

In process 34, CPU 10 then applies a wavelet transform to each of the image rows stored in line buffers 52, on a row-by-row basis. In other words, horizontal wavelet transform process 34 operates on the contents of each individual line buffer 52, independently from the contents of other line buffers 52. As discussed above, the particular wavelet transform executed in process 34 can be a wavelet transform of any type, including those in the above-incorporated U.S. Patents. A preferred wavelet transform is the Daubechies 9/7 transform described in the above-incorporated Antonini et al. article.

As a result of horizontal wavelet decomposition process 34, each input image row is decomposed into a low-frequency component and a high-frequency component. For an input image row of W pixels, each component is expressed as W/2 digital values, effectively decimating the data by a factor of 2. In addition, the result of the horizontal decomposition may be stored in the same memory space that stored the input image row. Figure 6b illustrates the result of process 34, after each of the K input image rows stored in line buffers 52 are decomposed. As shown in Figure 6b, line buffers 52_1 through 52_K each contain a low-frequency portion LOW FREQ in one half and a high-frequency portion HIGH FREQ in the other half.

Referring back to Figure 5, upon completion of process 34 for the initial K input image rows, vertical wavelet decomposition process 36 is next performed for one of the W columns of the K line buffers 52. Process 36 is performed on a single column m; in the first instance, of course, index m points to the first column of line buffers 52. Because the number K of line buffers 52 corresponds to the length K of the wavelet filter, as noted above, process 36 involves a single application of the low and high pass filter functions to the contents of the m^{th} column of line buffers 52. As known in the art, the application of the filter is made to an ordered arrangement of data values. According to the preferred embodiment of the invention, pointers 53 indicate this ordering. The one of line buffers 52 to which the pointer 53 having the value IPtr(1) points provides, in its m^{th} column, the first data value in the sequence to be filtered, the line buffer 52 associated with the pointer 53 having the value IPtr(2) provides the next data value in the sequence, and so on. Of course, in the first pass through process 36 (for each of the W columns), the value IPtr(1) is stored in the pointer 53 associated with line buffer 52_1, as shown in Figure 6b.

As noted above, each instance of process 36 returns two values, namely a low-frequency vertical component value and a high-frequency vertical component value. In process 38, CPU 10 stores these two values in an output buffer 54 (and not in line buffers 52, as the current contents of line buffers 52 will be needed, for the most part, for additional vertical decomposition passes). Output buffer 54 may reside either in a cache resource 12 or 14. However, since only two values are written for each pass through the process, output buffer 54 may be realized in a portion of main external memory 18, with minimal impact upon the process performance (especially if additional cache space is freed). Figure 6b illustrates the result of one instance of process 36, for the m^{th} column of line buffers 52 for the initial load of line buffers 52 (i.e., pointer value IPtr(1) pointing to line buffer 52_1). In the example of Figure 6b, the column index m is within the first half of line buffers 52, and thus processes 36, 38 in the illustrated instance are operating upon low-frequency horizontal component results. As shown in Figure 6b, two output results, low-frequency output result LL and high-frequency output result LH, are stored in output buffer 54; it will be understood by those in the art, of course, that for index m in the second half (i.e., m > W/2) of line buffers 52, the results of process 36 will correspond to HL and HH component values.

After storing process 38 of the two results from process 36, CPU 10 next evaluates decision 41 to determine whether additional columns remain to be vertically decomposed; in other words, decision 41 determines whether column index m is less than line buffer width W. If so (decision 41 is YES), additional columns remain to be processed. Column index value m is incremented in process 42, and control returns to vertical wavelet transform process 36 to decompose the next column of line buffers 52, and to store the results in output buffer 54 in process 38.

Upon the completion of the vertical decomposition of the current contents of line buffers 52 (i.e., m=W), decision 41 returns a NO result, and CPU 10 then evaluates decision 43 to determine whether the complete image has been decomposed. If not (decision 43 is NO), additional image rows remain to be loaded into line buffers 52 for decomposition in both directions. Accordingly, in process 44, CPU 10 scans the next two image rows into the two line buffers 52 pointed to by those pointers 53 having the pointer values IPtr(1) and IPtr(2). Effectively, since the pointer values IPtr(1), IPtr(2) point to the oldest and next oldest image rows, respectively, process 44 overwrites the data corresponding to these oldest image rows with the next image rows in sequence, retaining the results of the other rows.

CPU 10 then increments pointers 53 by two, in this exemplary embodiment, in process 46. As a result of process 46, the pointer 53 that previously had the value IPtr(3) now has the pointer value IPtr(1), the pointer 53 that previously stored the value IPtr(4) now stores IPtr(2), and so on. The pointer values IPtr() wrap around, such that the values of pointers 53 that point to the newly loaded image rows are now IPtr(K-1) and IPtr(K). Figure 6c illustrates the state of line buffers 52 and pointers 53 after the first instance of processes 44, 46, where the image rows K+1 and K+2 are first scanned into line buffers 52. The pointer 53 pointing to line buffer 52_1 has a value IPtr(K-1), and the pointer 53 pointing to line buffer 52_2 has a value IPtr(K). These two line buffers 52_1, 52_2 each contain image data, while the other line buffers 52_2 through 52_K continue to store the low and high frequency components from the previously performed horizontal wavelet transform, as shown in Figure 6c.

Once the new image rows are scanned into line buffers 52 in process 46, CPU 10 then performs a horizontal wavelet transform decomposition upon these two new rows, in process 48. Since the contents of the other line buffers 52 already contain horizontal decomposition results, process 48 is not performed upon the contents of these line buffers 52 (e.g. line buffers 52_2 through 52_K in Figure 6c). The horizontal wavelet transform of process 48 is of course the same transform as that performed in process 34. Column index m is then reset to point to the first column (m=1), in process 50, and control returns to vertical wavelet transform process 36 for the next vertical decomposition of each of the W columns of the current contents of line buffers 52, using the current values stored in pointers 53 to order the data. The results are written into the corresponding locations of output buffer 54.

CPU 10 then again performs processes 36 and 38 upon each column in the current contents of line buffers 52, with new image rows loaded into line buffers 52 and horizontally transformed in process 48 after all columns are complete. This operation continues until CPU 10 determines, in decision 43, that the entire image has been processed. If so (decision 43 is YES), the current level of wavelet decomposition process 22 is complete for the input image, and output buffer 54 contains the entire set of four decomposition components for the current decomposition level of the image.

Referring back to Figure 4, once horizontal and vertical wavelet decomposition process 22 is completed, CPU 10 then quantizes the high frequency components in process 24. These high-frequency components are retrieved from output buffer 54, either directly or indirectly. In decision 25, CPU 10 determines whether the desired compression level has been reached. If not (decision 25 is NO), process 22 is then repeated, using the LL component stored in output buffer 54 from the previously completed decomposition as the input image. Of course, since the coefficients of the LL component are decimated by a factor of two from level to level, the width W of line buffers 52 will be reduced accordingly for each decomposition level.

Upon the desired compression level being reached (decision 25), the wavelet-based image compression according to the preferred embodiment of the invention is complete, and the results are then encoded in process 26.

The wavelet decomposition process according to the preferred embodiment of the invention provides important advantages in efficiently and rapidly compressing digital image data. As evident from this description, the wavelet decomposition of the input image (as well as of low frequency components, for deeper decomposition levels) may be performed on the relatively small memory resources of line buffers 52. Because of these reduced memory requirements, according to modern technology, line buffers 52 may reside in the cache memory of the computer system, greatly speeding the compression process relative to conventional techniques in which main memory is used to store all image data and decomposition results. Specifically, the accesses to main memory are limited only to a single scanning of the image, row by row, into the line buffers, and to also possibly the writing of the decomposition results into main memory, should the output buffer be realized there. Further efficiency is provided by the rotation of pointer values to indicate the order in which the line buffer contents are to be vertically decomposed; this use of the pointers allows the contents of the line buffers to be retained in their same buffer locations as the wavelet filter is moved along in the vertical direction. The reduction in the number of overall memory accesses, in combination with the ability to effect the large majority of these accesses into cache rather than main memory, greatly improves the processing efficiency of the image compression process.

Referring now to Figures 7 and 8, encoding process 26 according to the preferred embodiment of the invention will now be described in detail. While the results of the previously-described compression process 22 may be encoded according to any encoding method, it is contemplated that the encoding method of Figures 7 and 8 is particularly beneficial; additionally, encoding process 26 described below may be applied to data processed according to any image transform process.

Referring to Figure 7, encoding process 26 begins with the receipt, in process 60, of an integer sequence from wavelet decomposition process 22, once decision 25 determines that the desired compression ratio has been reached. The received integer sequence is read (directly or indirectly) from output buffer 54, realized in cache memories 12, 14, or in main memory 18, as noted above. For purposes of this description, the received integer sequence X(m) contains N signed integers (i.e., each integer X(m) may be positive, negative, or zero), where m is an index of an integer within the sequence. Index m is initialized to 0 in process 62, as shown in Figure 7.

Decision 61 is next executed by CPU 10 of system 8 to analyze the mth integer X(m) in the received sequence; specifically, decision 61 determines whether integer X(m) is zero-valued. If not, control passes to decision 63, which determines whether integer X(m) is positive or negative. For positive integers X(m), decision 63 returns a YES result, and CPU 10 performs process 62 to generate an output integer Y(k) corresponding to integer X(m). In this embodiment of the invention, positive integers X(m) are scaled by a factor of two, so that:

$$Y(k) = 2 * X(m)$$

Index m is then incremented, also in process 62, and control passes to decision 65. For negative integers X(m), decision 63 returns a NO result. Process 64 is then performed by CPU 10 to convert the negative integer X(m) to an odd positive integer, adding one to the absolute value of X(m) after scaling by two:

$$Y(k) = (-2) * X(m) + 1$$

Also in process 64, index m is incremented, and control passes to decision 65.

Decision 65 determines whether additional integers X(m) in the received sequence remain to be processed, by comparing the newly incremented value of index m to length N. If additional integers X(m) remain (decision 65 is YES), control returns to decision 61 to test the next integer X(m), for the updated index m, against zero.

For an integer X(m) that has a zero value, decision 61 returns a YES result. Because encoding process 26 is being applied to the quantized results of a decomposition, it is expected that many integer values will be zero, with sequences of many contiguous zero-valued integers being likely. Accordingly, process 26 counts the number of sequential zero values, beginning with process 66 in which CPU 10 initializes a zero count C to zero, upon decision 61 detecting a first zero-valued integer X(m). In process 68, this zero count C is incremented, as is integer index m. The next integer X(m) in sequence is then tested for zero in decision 69; if X(m) is a zero (decision 69 is YES), zero count C and integer index m are both incremented in process 68, and decision 69 is repeated for the next integer X(m). This loop of decision 69 and count incrementing process 68 is repeated so long as integers X(m) in the retrieved sequence have zero values (or until the received integer sequence is exhausted).

Upon a non-zero value being detected in decision 69, CPU 10 performs process 70 to increment the zero count C by one, to C+1. This incrementing of zero count C to C+1 is performed to permit the saving of a byte in the encoded word for the maximum amount of cases. Incremented count C+1 is then converted into a binary representation, which of course consists of zeros and ones, and these zeros and ones are expanded into word lengths that are the same as that used to represent the non-zero integers. as will be described below. According to the preferred embodiment of the invention, in process 70, the leading "1" of the incremented binary count C+1 is not expanded or communicated, as its presence may be implied. By incrementing the zero count to C+1, a sequence of one zero integer may be encoded as a single byte of 0, and a sequence of two successive zero integers may be encoded as a single byte of 1. The minimum number of bytes required for encoding of zero sequences thus results from the incrementing of process 70.

Following the incrementing and expanding of process 70, decision 65 is then performed to determine whether the sequence has been completed; if not, control returns to decision 61 in which the non-zero integer X(m) is tested by decision 63, as discussed above.

According to this preferred embodiment of the invention, the converted output sequence Y(k) is comprised of a series of positive integers ranging from zero to the maximum value of the data word. In this output series, those integers ranging from two to the maximum value correspond to non-zero decomposition values, while those integers having zero and one value correspond to bits in the incremented count C+1 of consecutive zero values. The range of the integer value itself thus indicates the zero or non-zero nature of the encoded value, and a separate indication of zero or non-zero, such as provided in RLE and other conventional coding techniques, is not necessary.

An example of the operation of this encoding process will be instructive. Consider, by way of example, a sequence of bytes, each byte corresponding to a signed integer, as follows:

{+15, -3, 0, +9, +8, -33, 0, 0, 0, 0, . . . , 0, +43}

where the sequence of zeros numbers ninety zeros in succession. According to the preferred embodiment of the invention, the positive integers +15, +9, +8, +43 would be doubled in value to +30, +18, +16, +86, respectively, and the negative integers -3, -33 would be converted to the positive integers +7, +67, respectively. These doubled values could still be represented within a byte, in this example, as the sign bit is no longer necessary (permitting its use as a magnitude bit). The single zero has an incremented zero count C+1 of two (10_2), and is thus expanded to a single 0 byte, with the leading "1" bit implied as discussed above. The ninety zeros in succession, according to the preferred embodiment of the invention, would result in an incremented zero count C-1 of ninety-one which, converted to binary, is 1011011_2. Accordingly, the ninety zeros would be coded into a six-byte sequence {0, 1, 1, 0, 1, 1}, with the leading "1" implied. None of these values would be confused with a non-zero integer value, as all non-zero integer values are scaled to be no smaller than two. According to this example, therefore, the output sequence Y(k) is:

$$\{30, 7, 0, 18, 16, 67, 0, 1, 1, 0, 1, 1, 86\}$$

The input sequence of ninety-seven bytes is thus compressed to thirteen bytes, in this example.

This compression of the integer sequence is fully reversible. Upon receipt of a communicated compressed image, or in the process of displaying a stored compressed image, the computer system will first reverse such encoding and modulation that was applied to the compressed integer sequence prior to storage or translation, including the split sequence coding described below, to provide the sequence of integers Y(k) noted above. The decompression will then continue by analyzing each data word to determine its integer value, and converting the integer to the appropriate signed value and expanding the sequences of zero values, accordingly. For even-valued integer values, the decompression will divide the integer value Y(k) by two to return the positive signed integer X(m). For odd-valued integer values (greater than one), the decompression will subtract one and divide by negative two, returning a negative signed integer X(m). Upon detecting an integer value Y(k) of one or zero, the decompression routine will retrieve the next integer values in sequence that are also one or zero, decode the binary value represented by that sequence, and generate a sequence of zeroes corresponding to the decoded binary value minus one. According to this preferred embodiment of the invention, the resulting sequence X(m) will then be reconstructed by reversing the wavelet decomposition described above.

Referring back to Figure 7, the output byte sequence Y(k) is then further encoded, in process 72, as desired in the particular implementation. According to the preferred embodiment of the invention, encoding process 72 is carried out according to a split sequence coding, as will now be described relative to Figure 8.

Split-sequence coding process 72 begins with the receipt of a series of n-bit integers, in process 74. In process 76, CPU 10 then divides this bitstream of n-bit integers into blocks, which in this example are sixteen bytes in length. The further encoding of process 72 is then carried out on a block basis, beginning with process 78 in which CPU 10 sums the integers contained within a given block, and compares the value of this sum with a set of predetermined sum categories, which are effectively ranges of possible sum values. In this example, eight sum categories are established, with category 0 corresponding to the lowest sum values and category 7 corresponding to the highest. The boundaries of sum values may be uniformly set over the entire range, or may alternatively be selected according to a desired dynamic range.

The generation of an output code value, for each integer in the received bitstream, begins with process 80 in which a code for the sum category of its block is output. For this example, in which eight sum categories are established, a three-bit code k is output for the first integer i in the block, indicating the range of the block sum. In process 82, the least significant bits of integer i are shifted out and appended to the output bitstream; in this split sequence embodiment of the invention, k LSBs are shifted out from integer i, where k is the index of the sum category into which its block sum fell. Accordingly, the number of bits shifted out in process 82 depends upon the average value of the integers in the block, with fewer bits shifted out for relatively low level integer values and vice versa.

In process 84, the remaining bits of integer i are then encoded into a sequence of zeroes, followed by a trailing one, with the number of zeros corresponding to the binary value represented by the remaining n-k bits. This sequence of zeros and a trailing one are appended to the output bitstream, completing the encoding of integer i in the current block.

An example of this split sequence encoding process 72 will be instructive. Consider, for example, an integer i having a value of 54_{10} (00110110_2), resident in an integer block having a sum in category 4 (resulting from the calculation of process 78). In process 80, the first three bits output for integer i will thus be 100_2, corresponding to sum category 4. Accordingly, the value k of 4 causes four LSBs to be shifted out of integer i in process 82, appending the value 0110_2 to the bitstream. The remaining n-k bits 0011_2 are then encoded, in process 84, into three 0's followed by a trailing 1, corresponding to the remaining binary value of three. Accordingly, the encoding of the value 54 results in a binary word of:

100 0110 0001

In general, therefore, additional compression is thus provided by process 72, in a fully reversible manner. Process 72 is particularly beneficial when applied to data generated by the wavelet compression and encoding methods of the preferred embodiment of the invention, because the prevalence of small numbers (integers of zero and one values) are very efficiently encoded by the split sequence encoding of process 72.

According to this preferred embodiment of the invention, encoding process 26 provides a high level of data compression to the quantized results of the image decomposition. This encoding is relatively simple to implement, and is well-suited for use with wavelet-decomposed information.

Referring back to Figure 4, the image compression process according to the preferred embodiment of the invention is then completed by system 8 performing process 28 to store the encoded compressed results in its memory, such as main memory 18 or disk storage device 19. Depending upon the eventual destination of the compressed image, transmission of the compressed image over a network, such as a local area network (LAN) or the Internet, may then be carried out. In the event of such transmission, the receiving computer system would, of course, reconstruct the image by reversing the encoding processes described above, followed by wavelet reconstruction.

The present invention provides numerous important advantages in the compression of digital images, particularly in the case of still images. Regarding the wavelet decomposition of the present invention, accesses to main memory are reduced significantly, as the wavelet decomposition may be carried out in a relatively small amount of memory, such as may be realized in cache memory. As a result, image compression according to the present invention is particularly efficient both in terms of memory utilization and also system performance.

Additionally, the encoding techniques according to the present invention can provide additional efficiency in the storage and communication of digital images. In particular, long sequences of zeros may be efficiently encoded among a sequence of non-zero data, without requiring an additional field to indicate which values pertain to the zero sequence.

While the present invention has been described according to its preferred embodiments, it is of course contemplated that modifications of, and alternatives to, these embodiments, such modifications and alternatives obtaining the advantages and benefits of this invention, will be apparent to those of ordinary skill in the art having reference to this specification and its drawings. It is contemplated that such modifications and alternatives are within the scope of this invention as subsequently claimed herein.

I CLAIM:

1. A method of compressing a digital representation of an image, comprising the steps of:

loading each of a plurality of line buffers with a row of the image, each line buffer having a number of entries corresponding to columns of the image, the number of line buffers in the plurality of line buffers corresponding to a filter length;

for each of the plurality of line buffers, performing a filter decomposition along the line buffer to produce a low-frequency portion and a high-frequency portion, and storing the low-frequency and high-frequency portions in the line buffer;

then performing a filter decomposition upon the contents of the columns of the plurality of the line buffers to produce, for each of the columns, a low-frequency component value and a high-frequency component value;

storing the low-frequency component value and the high-frequency component value in an output buffer;

then loading a subset of the plurality of line buffers with next rows of the image, the subset of the plurality of line buffers corresponding to the rows of the image furthest from the next rows of the image;

for each of the subset of the plurality of line buffers, performing a filter decomposition along the line buffer to produce a low-frequency portion and a high-frequency portion, and storing the low-frequency and high-frequency portions in the line buffer; and

repeating the performing, storing, loading, performing, and repeating steps over the image.

2. The method of claim 1 wherein, upon completion of the repeating step over the image, the output buffer contains a plurality of components of the image, comprising a low-low component, a high-low component, a low-high component, and a high-high component.

3. The method of claim 2, further comprising:

loading each of the plurality of line buffers with a row of the low-low component;

for each of the plurality of line buffers, performing a filter decomposition along the line buffer to produce a low-frequency portion and a high-frequency portion, and storing the low-frequency and high-frequency portions in the line buffer;

then performing a filter decomposition upon the contents of the columns of the plurality of the line buffers to produce, for each of the columns, a low-frequency component value and a high-frequency component value;

storing the low-frequency component value and the high-frequency component value in the output buffer;

then loading a subset of the plurality of line buffers with next rows of the low-low component;

for each of the subset of the plurality of line buffers, performing a filter decomposition along the line buffer to produce a low-frequency portion and a high-frequency portion, and storing the low-frequency and high-frequency portions in the line buffer; and

repeating the performing, storing, loading, performing, and repeating steps over the image.

4. The method of claim 1, further comprising:

before the step of performing a filter decomposition upon the contents of the columns of the plurality of the line buffers, assigning pointer values to a plurality of pointers, each pointer associated with one of the line buffers, wherein the pointer values correspond to the positions, in the image, of the rows stored in the associated line buffers; and

wherein the step of performing a filter decomposition upon the contents of the columns of the plurality of the line buffers applies a filter to the ordered contents of the columns, the ordering corresponding to the pointer values.

5. The method of claim 4, further comprising:

selecting the subset of the plurality of line buffers to be loaded in the loading step according to the pointer values.

6. The method of claim 5, further comprising:

in association with the step of loading a subset of the plurality of line buffers with next rows of the image, advancing the pointer values.

7. The method of claim 6, wherein the selecting step is performed by evaluating the pointer values to determine those line buffers associated with the oldest image data.

8. The method of claim 1, wherein the filter decomposition is a wavelet filter decomposition.

9. The method of claim 1, wherein, upon completion of the repeating step over the image, the output buffer contains a plurality of components of the image,

and further comprising:

after the repeating step, quantizing the components of the image.

10. The method of claim 9, further comprising:

arranging the quantized components of the image into a sequence of integers;

scaling each of the sequence of integers that has a non-zero value to a range above a minimum integer value greater than one;

counting the number of successive ones of the sequence of integers that have a zero value to derive a zero count value;

expressing the zero count value as a binary value;

expanding the binary zero count value into a sequence of data words of zero and one values; and

generating an encoded bitstream corresponding to the scaled sequence of integers and the sequence of data words of zero and one values.

11. The method of claim 10, wherein the sequence of integers comprises a sequence of signed integers;

and wherein the scaling step comprises:

for each of the sequence of integers that has a positive value, doubling the positive value; and

for each of the sequence of integers that has a negative value, doubling the absolute value of the negative value and adding one.

12. The method of claim 10, wherein the zero count value corresponds to the number of successive ones of the sequence of integers that have a zero value, plus one.

13. A system for compressing digital image data representative of an image, comprising:

a source of the image data;

a main memory for storing the image data;

a cache memory; and

a central processing unit, coupled to the source, to the main memory , and to the cache memory, programmed to perform the operations of:

loading each of a plurality of line buffers in the cache memory with a row of the image data, each line buffer having a number of entries corresponding to columns of the image data, the number of line buffers in the plurality of line buffers corresponding to a filter length;

for each of the plurality of line buffers, performing a filter decomposition along the line buffer to produce a low-frequency portion and a high-frequency portion, and storing the low-frequency and high-frequency portions in the line buffer;

then performing a filter decomposition upon the contents of the columns of the plurality of the line buffers to produce, for each of the columns, a low-frequency component value and a high-frequency component value;

storing the low-frequency component value and the high-frequency component value in an output buffer region of memory;

then loading a subset of the plurality of line buffers with next rows of the image data, the subset of the plurality of line buffers corresponding to the rows of the image furthest from the next rows of the image;

for each of the subset of the plurality of line buffers, performing a filter decomposition along the line buffer to produce a low-frequency portion and a high-frequency portion, and storing the low-frequency and high-frequency portions in the line buffer; and

repeating the performing, storing, loading, performing, and repeating steps over the image data.

14. The system of claim 10, wherein the output buffer is in main memory.

15. A method of encoding quantized digital data, comprising the steps of:

arranging the quantized data of the image into a sequence of integers;

scaling each of the sequence of integers that has a non-zero value to a range above a minimum integer value greater than one;

counting the number of successive ones of the sequence of integers that have a zero value to derive a zero count value;

expressing the zero count value as a binary value;

expanding the binary zero count value into a sequence of data words of zero and one values; and

generating an encoded bitstream corresponding to the scaled sequence of integers and the sequence of data words of zero and one values.

16. The method of claim 15, wherein the sequence of integers comprises a sequence of signed integers;

and wherein the scaling step comprises:

for each of the sequence of integers that has a positive value, doubling the positive value; and

for each of the sequence of integers that has a negative value, doubling the absolute value of the negative value and adding one.

17. The method of claim 15, wherein the zero count value corresponds to the number of successive ones of the sequence of integers that have a zero value, plus one.

18. The method of claim 17, wherein the step of expressing the zero count value as a binary value comprises:

expressing the zero count value as a binary number having a leading "1" bit; and

striking the leading "1" bit from the binary number to produce the binary value.

19. The method of claim 15, wherein the generating step comprises:

arranging the scaled sequence of integers and the sequence of data words into blocks;

summing the values of each of the blocks;

associating each sum from the summing step with one of a plurality of sum categories, each sum category having a digital value associated therewith that corresponds to the relative magnitude of its sums;

for each integer within a block, generating an encoded data word comprising:

a first portion corresponding to the digital value of the sum category of the block;

an LSB portion corresponding to a selected number of least significant bits of the integer, the selected number corresponding to the digital value of the sum category of the block; and

an encoded MSB portion corresponding to remaining bits of the integer not included in the selected number of least significant bits.

20. The method of claim 19, wherein the encoded portion of the remaining bits corresponds to a sequence of zero bits numbering the digital value of the remaining bits, followed by a trailing one bit.

* * * * *

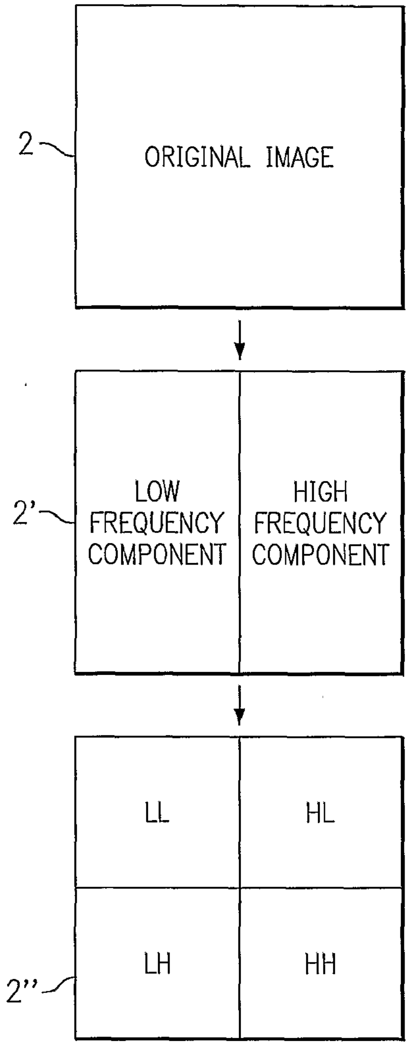


FIG. 1
(PRIOR ART)

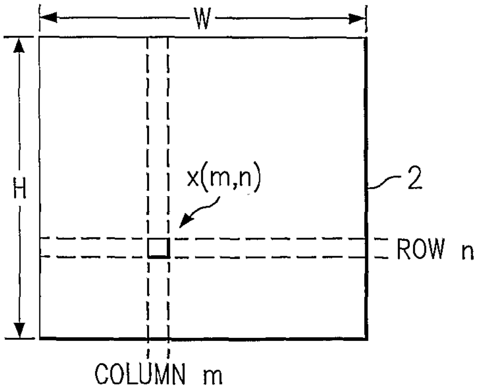


FIG. 2
(PRIOR ART)

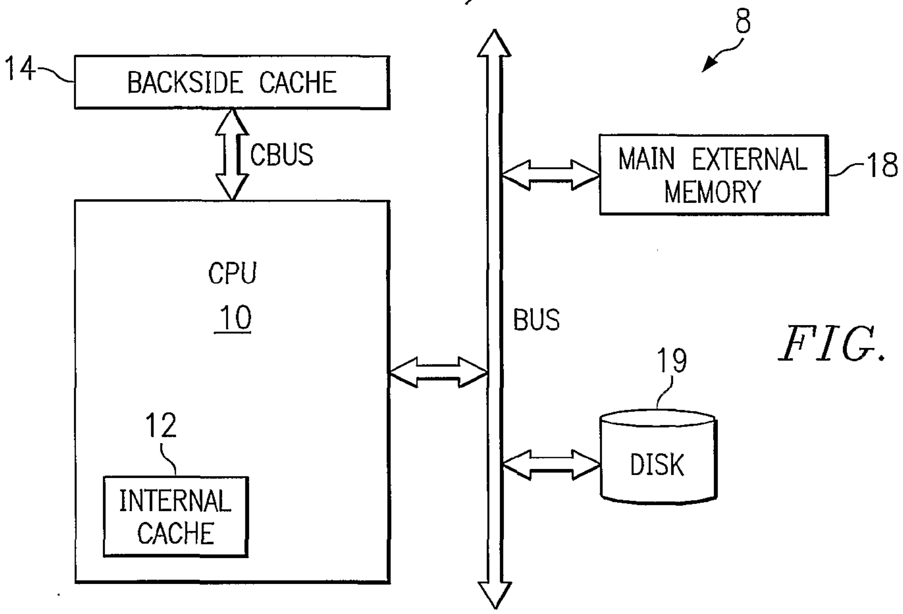

8
14
BACKSIDE CACHE
CBUS
CPU
10
12
INTERNAL CACHE
BUS
MAIN EXTERNAL MEMORY
18
19
DISK

FIG. 3

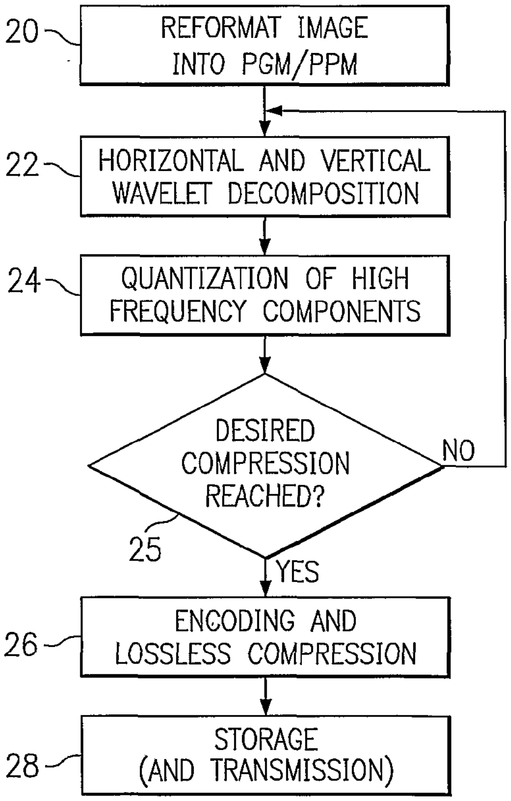

20
REFORMAT IMAGE INTO PGM/PPM
22
HORIZONTAL AND VERTICAL WAVELET DECOMPOSITION
24
QUANTIZATION OF HIGH FREQUENCY COMPONENTS
DESIRED COMPRESSION REACHED?
NO
25
YES
26
ENCODING AND LOSSLESS COMPRESSION
28
STORAGE (AND TRANSMISSION)

FIG. 4

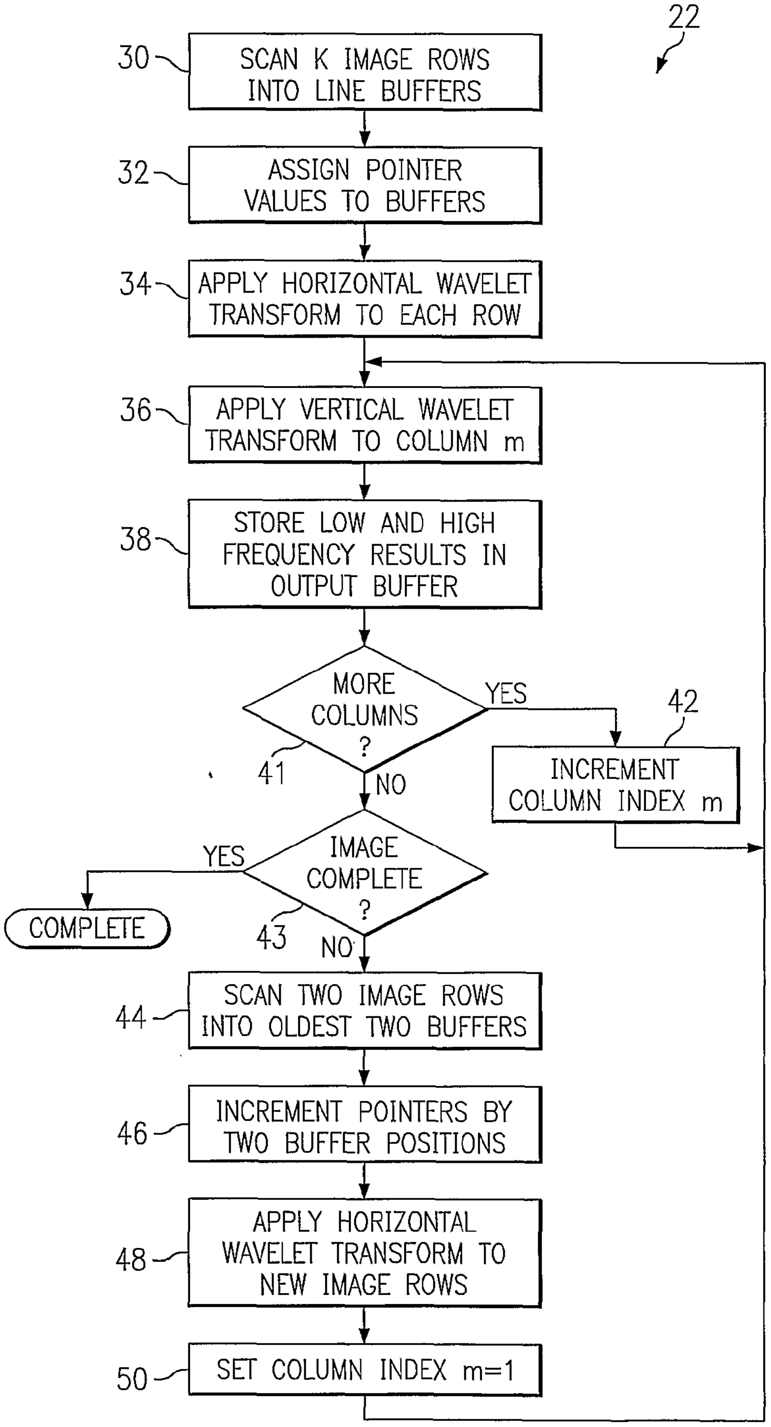

22
30
SCAN K IMAGE ROWS INTO LINE BUFFERS
32
ASSIGN POINTER VALUES TO BUFFERS
34
APPLY HORIZONTAL WAVELET TRANSFORM TO EACH ROW
36
APPLY VERTICAL WAVELET TRANSFORM TO COLUMN m
38
STORE LOW AND HIGH FREQUENCY RESULTS IN OUTPUT BUFFER
MORE COLUMNS ?
41
YES
42
INCREMENT COLUMN INDEX m
NO
IMAGE COMPLETE ?
43
YES
COMPLETE
NO
44
SCAN TWO IMAGE ROWS INTO OLDEST TWO BUFFERS
46
INCREMENT POINTERS BY TWO BUFFER POSITIONS
48
APPLY HORIZONTAL WAVELET TRANSFORM TO NEW IMAGE ROWS
50
SET COLUMN INDEX m=1

FIG. 5

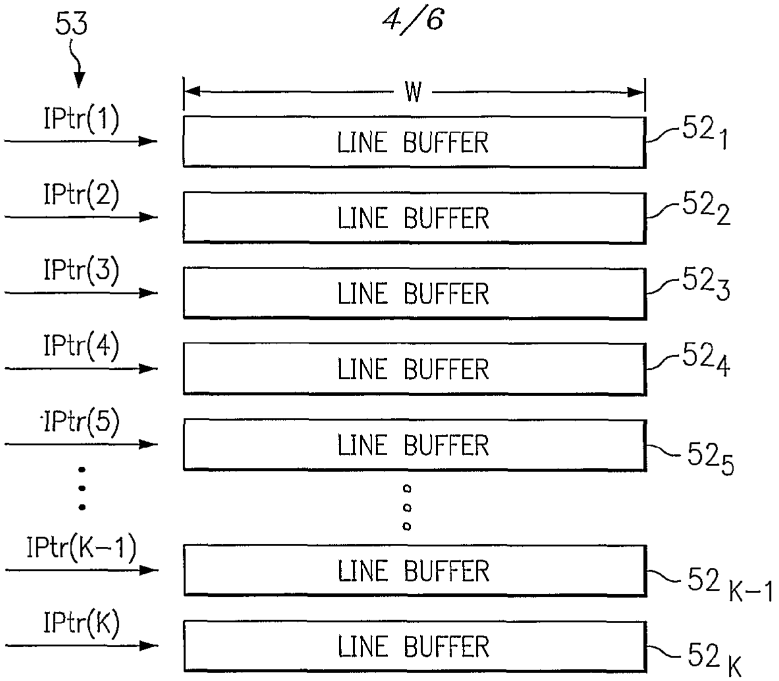

53
W
IPtr(1)
LINE BUFFER
52_1
IPtr(2)
LINE BUFFER
52_2
IPtr(3)
LINE BUFFER
52_3
IPtr(4)
LINE BUFFER
52_4
IPtr(5)
LINE BUFFER
52_5
IPtr(K−1)
LINE BUFFER
52_{K-1}
IPtr(K)
LINE BUFFER
52_K

FIG. 6a

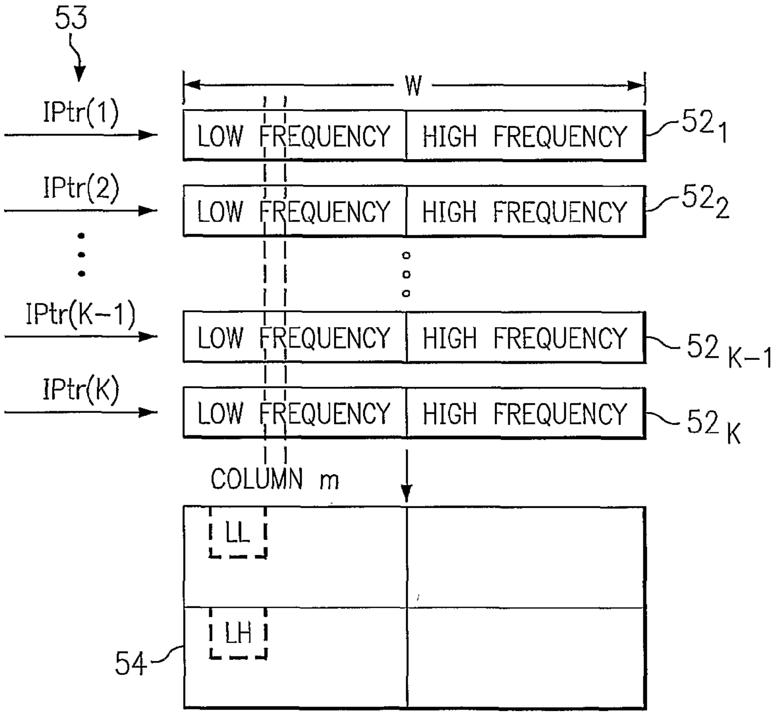
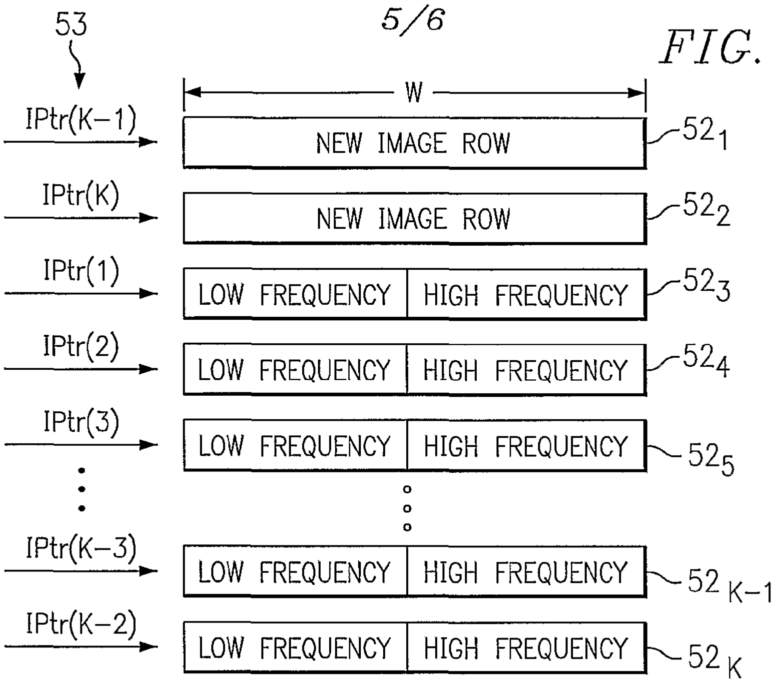

53
W
IPtr(1)
LOW FREQUENCY
HIGH FREQUENCY
52_1
IPtr(2)
LOW FREQUENCY
HIGH FREQUENCY
52_2
IPtr(K−1)
LOW FREQUENCY
HIGH FREQUENCY
52_{K-1}
IPtr(K)
LOW FREQUENCY
HIGH FREQUENCY
52_K
COLUMN m
LL
LH
54

FIG. 6b

FIG. 6c


53
W
IPtr(K−1)
NEW IMAGE ROW
52_1
IPtr(K)
NEW IMAGE ROW
52_2
IPtr(1)
LOW FREQUENCY
HIGH FREQUENCY
52_3
IPtr(2)
LOW FREQUENCY
HIGH FREQUENCY
52_4
IPtr(3)
LOW FREQUENCY
HIGH FREQUENCY
52_5
IPtr(K−3)
LOW FREQUENCY
HIGH FREQUENCY
52_{K-1}
IPtr(K−2)
LOW FREQUENCY
HIGH FREQUENCY
52_K

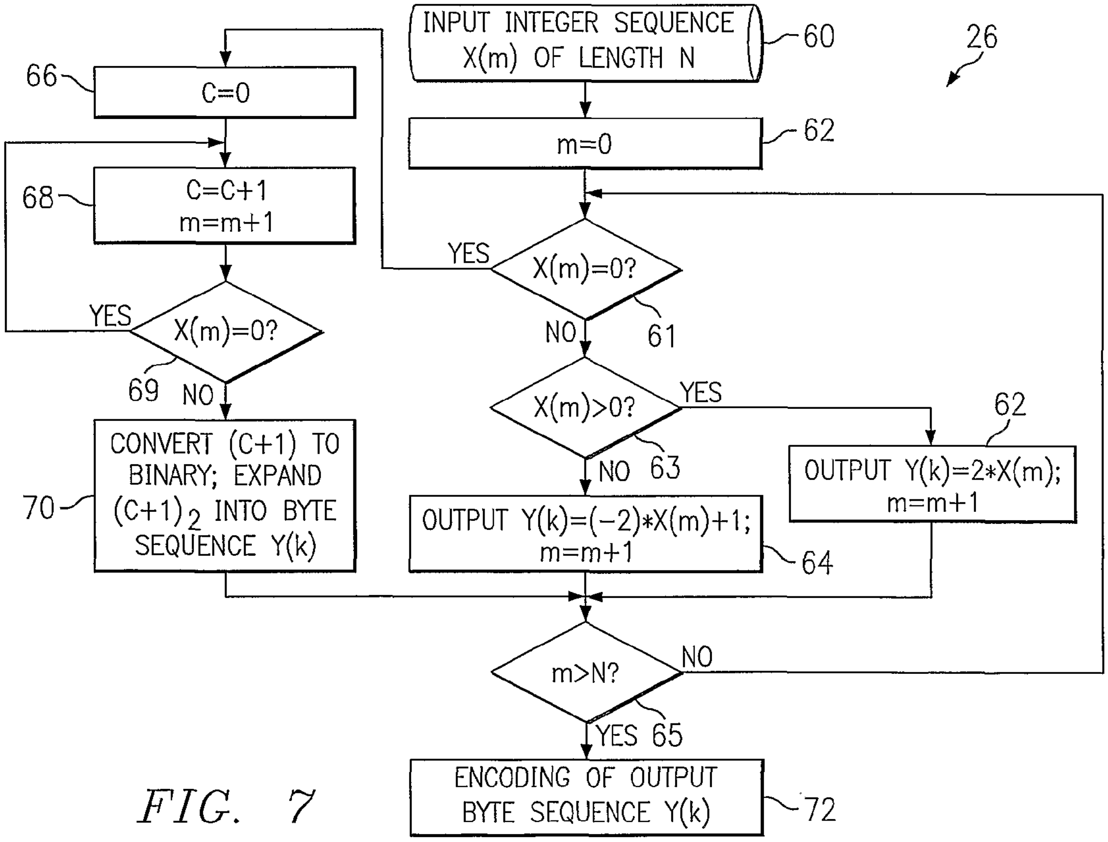

INPUT INTEGER SEQUENCE X(m) OF LENGTH N
60
26
m=0
62
X(m)=0?
61
YES
NO
66
C=0
68
C=C+1
m=m+1
X(m)=0?
69
YES
NO
70
CONVERT (C+1) TO BINARY; EXPAND $(C+1)_2$ INTO BYTE SEQUENCE Y(k)
X(m)>0?
63
YES
NO
62
OUTPUT Y(k)=2*X(m); m=m+1
OUTPUT Y(k)=(−2)*X(m)+1; m=m+1
64
m>N?
65
NO
YES
ENCODING OF OUTPUT BYTE SEQUENCE Y(k)
72

FIG. 7

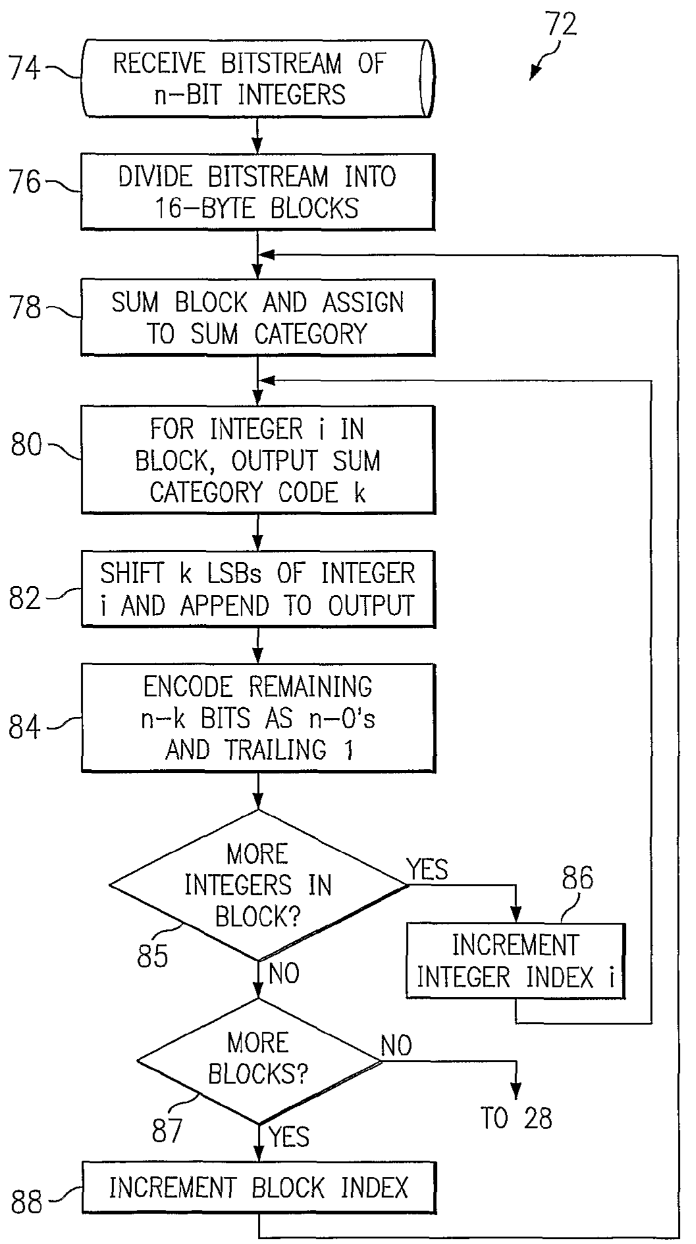

72
74
RECEIVE BITSTREAM OF n-BIT INTEGERS
76
DIVIDE BITSTREAM INTO 16-BYTE BLOCKS
78
SUM BLOCK AND ASSIGN TO SUM CATEGORY
80
FOR INTEGER i IN BLOCK, OUTPUT SUM CATEGORY CODE k
82
SHIFT k LSBs OF INTEGER i AND APPEND TO OUTPUT
84
ENCODE REMAINING n-k BITS AS n-0's AND TRAILING 1
MORE INTEGERS IN BLOCK?
85
YES
86
INCREMENT INTEGER INDEX i
NO
MORE BLOCKS?
87
NO
TO 28
YES
88
INCREMENT BLOCK INDEX

FIG. 8

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US01/16579

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) : G06K 9/36; HO4N 7/12

US CL : 382/232, 233, 236, 239, 240, 246, 248; 348/398, 407

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 382/232, 233, 236, 239, 240, 246, 248; 348/398, 407

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,604,824 A (CHUI et al.) 18 FEBRUARY 1997, SEE ABSTRACT; FIGURES 1-2, 4-21; AND COLUMN 1, LINE 1 TO COLUMN 6, LINE 10.	1-20
Y	US 5,748,116 A (CHUI et al.) 05 MAY 1998, SEE ABSTRACT; FIGURES 1-5; AND COLUMN 1, LINES 1-10 & COLUMN 1 LINE 48 TO COLUMN 3 LINE 38.	1-20
Y	US 5,949,911 A (CHUI et al.) 07 SEPTEMBER 1999, SEE ABSTRACT; FIGURES 2-7; AND COLUMN 1, LINE 1 TO COLUMN 2, LINE 10.	4-7, 9-12, & 14-20

□ Further documents are listed in the continuation of Box C. □ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier application or patent published on or after the international filing date		
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed	"&"	document member of the same patent family

Date of the actual completion of the international search

09 July 2001 (09.07.2001)

Date of mailing of the international search report

26 JUL 2001

Name and mailing address of the ISA/US

Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231

Facsimile No. (703)305-3230

Authorized officer For

Jose L. Couso

Kugenia Logan

Telephone No. 7033053800